Exploiting Monge Structures in Optimum Subwindow Search

Senjian An, Patrick Peursum, Wanquan Liu, Svetha Venkatesh and Xiaoming Chen
Dept. of Computing, Curtin University of Technology
GPO Box U1987, Perth, WA 6845, Australia.
s.an,p.peursum, w.liu, s.venkatesh,x.chen@curtin.edu.au

Abstract

Optimum subwindow search for object detection aims to find a subwindow so that the contained subimage is most similar to the query object. This problem can be formulated as a four dimensional (4D) maximum entry search problem wherein each entry corresponds to the quality score of the subimage contained in a subwindow. For $n \times n$ images, a naive exhaustive search requires $O(n^4)$ sequential computations of the quality scores for all subwindows. To reduce the time complexity, we prove that, for some typical similarity functions like Euclidian metric, χ^2 metric on image histograms, the associated 4D array carries some Monge structures and we utilise these properties to speed up the optimum subwindow search and the time complexity is reduced to $O(n^3)$. Furthermore, we propose a locally optimal alternating column and row search method with typical quadratic time complexity $O(n^2)$. Experiments on PASCAL VOC 2006 demonstrate that the alternating method is significantly faster than the well known efficient subwindow search (ESS) method whilst the performance loss due to local maxima problem is negligible.

1. Introduction

Object detection and localization aims to find objects in images and has become a hot topic in computer vision in recent years. Since natural objects have complex shapes, it is hard to find the exact position and the shape of the objects in images. An easier way is to find the bounding box of the object and this method is called sliding window methods [6, 11, 17, 12]. The first step of sliding window methods is to train a quality function based on the extracted SIFT[16], SURF [5] or other type of features from training images, and the next is to apply the quality function on all possible sub-images to find the object by maximizing the quality scores. A naive exhaustive search is to apply the quality function on all $O(n^4)$ subwindows in $n \times n$ images. This may be computationally prohibitive in practice. To reduce the computational complexity, many approximate methods were proposed [8, 9, 22, 21]. Recently, a fast globally optimal method, called Efficient Subwindow Search (ESS) [14, 15], was developed to solve this problem. Since ESS is a branch and bound method and the upper bound on the number of its iteration is $O(n^4)$, the running time has a wide variance across images and the worst case time complexity is $O(n^4)$, close to the exhaustive search. In [3], a faster branch and bound method, called improved ESS (I-ESS), is proposed. The worse case complexity of I-ESS is cubic and it is much faster than ESS in the experiments on PASCAL VOC 2006. To further speed up the subwindow search, [3] also proposed a locally optimal method, called A-ESS. However, I-ESS and A-ESS are only applicable for the case when the quality function is linear on the histogram of the extracted features. In this case, the quality score of a subwindow simply sums up the contributions of the extracted features [14, 3]. This allows the subwindow search problem to be reduced to a maximum submatrix problem that finds a region in a matrix with the largest sum of entries. In the case when the quality function is nonlinear, the subwindow search is no longer a maximum submatrix search problem and A-ESS and I-ESS are not applicable.

Since nonlinear classifiers using kernels usually outperform linear methods [21], it is important to develop efficient algorithms for optimum subwindow search with nonlinear quality functions. Although ESS can be applied for some nonlinear quality functions [14, 15], its performance has large variances across images as in the linear case. In this paper, we examine techniques to find computationally efficient techniques with lesser variance across images, thus leading to stable performance. To do so, we formulate the optimal sub-window search problem as a four dimensional combinatorial optimisation problem, which finds the maximum entry in array, wherein each entry corresponds to the quality score of the sub-image contained in a subwindow. Our theoretical contribution is to show that if the quality function satisfies certain conditions, the four dimensional array carries nice Monge properties [7] which has been widely used to speed up in many optimization problems. We utilise these properties to formulate new algo-

978-1-4244-6983-3/10/$26.00 ©2010 IEEE

rithms for optimum subwindow search problem to reduce the time complexity to $O(n^3)$. Furthermore, we propose a locally optimal method, called ARCS, to further reduce the complexity to $O(n^2)$. Though ARCS is based on alternating optimization and may encounter local maxima problems, our experiments demonstrate that the performance loss due to local maxima problem is negligible whilst being significantly faster than ESS. The significance of our approach is to identify the utility of using Monge properties in the context of computer vision tasks, and the proposed paradigm has potential to be used in more general settings.

The rest of the paper is organized as follows. Section 2 reviews the Monge property of matrices and the efficient algorithm to find the row maxima. The row maxima can be found by visiting $O(n)$ number of matrix elements. In section 3 we will formulate the optimum search algorithm as an 4D array search problem and address the array's Monge properties. Section 4 addresses the alternating search method and Section 5 the application issues in objection detection. The experimental results are provided in Section 6.

2. Monge Matrices

An $m \times n$ real matrix $A = \{a[i, j]\}$ is called a *Monge Matrix* if it satisfies the so-called *Monge Property*

$$a[i, j] + a[k, l] \leq a[i, l] + a[k, j], \forall i < k, j < l. \quad (1)$$

A is called *Inverse Monge Matrix* if the above inequalities hold in the reverse direction, i.e.,

$$a[i, j] + a[k, l] \geq a[i, l] + a[k, j], \forall i < k, j < l. \quad (2)$$

There are many interesting and useful properties for (invese) Monge matrices. We only present the following three fundamental properties that will be used in the following sections. The first two are useful in checking whether a matrix is Monge and the third is related to an efficient algorithm for computing the row maxima, fundamental to our application.

Proposition 1 *[7] The Monge property or Inverse Monge Property holds if and only if it holds for adjacent rows and adjacent columns, that is, it suffices to require*

$$a[i, j] + a[i + 1, j + 1] \leq a[i, j + 1] + a[i + 1, j], \quad \forall 1 \leq i < m, 1 \leq j < n \quad (3)$$

for real Monge matrices, and

$$a[i, j] + a[i + 1, j + 1] \geq a[i, j + 1] + a[i + 1, j], \quad \forall 1 \leq i < m, 1 \leq j < n \quad (4)$$

for real inverse Monge matrices.

For a Monge matrix A, if we reverse the order of its rows, then (4) will be satisfied. So we have

Corollary 1 *By reversing the order of its rows, a Monge matrix will become an inverse Monge matrix and vise versa.*

Proposition 2 *[7] The class of $m \times n$ (inverse) Monge matrices forms a convex cone in the vector space of all real matrices with same dimensions. That is, if A and B are real (inverse) Monge matrices, then $A + B$ and λA are also (inverse) Monge matrices where $\lambda \geq 0$.*

Let A be an $m \times n$ matrix and let $j(i)$ be the column of A which contains the leftmost maximum entry in row i. If $j(i)$ is non-decreasing, i.e. $j(1) \leq j(2), \cdots, \leq j(m)$, we call A monotone. If all its submatrices are monotone, we call A totally monotone. For a totally monotone matrix A, all its row maxima can be computed in $O(n + m)$ time by the so-called SMAWK algorithm [1].

Proposition 3 *[7] Inverse Monge matrices are totally monotone matrices, whose row maxima can be computed in $O(n + m)$ time.*

By Corollary 1, a Monge matrix is an inverse Monge matrix by reversing its row order. Following Proposition 3, we have

Corollary 2 *[7] The row maxima of Monge matrices can be computed in $O(n + m)$ time.*

Also, there is an efficient parallel algorithm to compute the row maxima of Monge matrices.

Proposition 4 *[4] Inverse Monge matrices are totally monotone matrices whose row maxima can be computed in $O(\log n)$ time with $O(n)$ processors in CREW-PRAM or EREW-PRAM model.*

EREW-PRAM is the parallel model where the processors operate synchronously and share a common memory, but no two processors are allowed simultaneous access to a memory cell (whether the access is for reading or writing in that cell). The CREW-PRAN differs from EREW-PRAM in that simultaneous reading is allowed but simultaneous writing is still forbidden.

In the next section, we will show that the optimum subwindow search problem can be formulated as a maximum entry search problem of 4D arrays wherein each 2D subarray with fixed row (or column) range is an inverse Monge matrix.

3. Optimum Subwindow Search

In object localization or image part retrieval, sliding window methods have been widely applied. Sliding window methods first extract features from training images and cluster these features into K cluster bins. Then for any image, one counts the number of features in each bin, and formulate these counts as a vector $\mathbf{h}$, which is often called a histogram. Based on the histogram of the training images, sliding window methods train a quality function for each object and then apply the quality function on all possible sub-images to find the object by maximizing the quality scores. In the following, we use $f(\mathbf{h})$ to denote the quality function.

For an $m \times n$ image, let us use a 3-dimensional (3D) array $H = \{h[i,j,k]\}$ to represent the histogram at each pixel, i.e., $h[i,j,k]$ denotes the count of k-th bin feature extracted at pixel (i,j). The histogram of the sub-image within subwindow $w = [t:b, l:r]$ can be computed as

$$\mathbf{h}_w(k) = \sum_{i=t}^{b}\sum_{j=l}^{r} h(i,j,k) \tag{5}$$

where t, b, l, r denote the top, bottom, left and right boundary of the subwindow w.

The sliding window methods aim to find a subwindow $w = [t:b, l:r]$ so that $f(\mathbf{h}_w)$ is maximum [14], that is

$$\max_{w \in \mathcal{W}} f(\mathbf{h}_w) \tag{6}$$

where $\mathcal{W}$ is the set of all possible subwindows, i.e.,

$$\mathcal{W} = \{w = [t:b, l:r] | 1 \le t \le b \le m, 1 \le l \le r \le n\}. \tag{7}$$

We will call (6) the optimum subwindow search problem hereafter. Now let us define a 4D array A as

$$A[t,b,l,r] = \begin{cases} f(\mathbf{h}_w); & \text{if } w = [t:b, l:r] \in \mathcal{W} \\ -\infty, & \text{otherwise} \end{cases} \tag{8}$$

Then the optimum subwindow search problem is transformed as a maximum entry search problem in a 4D array, i.e.,

$$\max_{[t:b,l:r] \in \mathcal{W}} A[t,b,l,r]. \tag{9}$$

An exhaustive search needs to compute all its $O(n^4)$ entries. However, for some typical quality functions $f(\mathbf{h})$, the 4D array A carries some Monge structures that can be used to speed up the optimum subwindow search. Now we present the main theoretical result in this paper. The proof is delegated to the appendix.

Theorem 1 *Assume that the quality function can be decomposed as*

$$f(\mathbf{h}) = \sum_k f_k\{\mathbf{h}(k)\} \tag{10}$$

and define the 4D array A as in (8). If all $f_k(x)$ are concave in $[0, +\infty]$, i.e.

$$f_k''(x) \le 0, \forall\, x \ge 0, k = 1, 2, \cdots, K, \tag{11}$$

then the 2D subarrays $A(\cdot,\cdot,l,r)$ are inverse Monge matrices for every fixed pair (l,r). Similarly, $A(t,b,\cdot,\cdot)$ are inverse Monge matrices for every fixed pair (t,b).

Here, we used $A(\cdot,\cdot,l,r)$ (and $A(t,b,\cdot,\cdot)$) to represent a 2D subarray of A with given (l,r) (and (t,b) respectively).

From Theorem 1 and Proposition 3, for each fixed (t,b), the optimal (l,r) can be found in $O(n)$ time by using SMAWK algorithm. Hence, we have the following Corollary of Theorem 1:

Corollary 3 *If the quality function satisfies the condition in Theorem 1, the maximum entry of A and thus the optimum subwindow can be found by visiting its $O(m^2n)$ entries with time complexity $O(m^2nK)$.*

The time complexity is still too high to be applied in practice and we propose an alternating optimization method in the next section to further reduce the computations. Before we move to next section, we give some examples where the quality function satisfies the conditions in Theorem 1.

3.1. χ^2 Metric

In computer vision, the similarity between two images is typically measured by χ^2 distance of their bag-of-visual-words histograms $\mathbf{h}^Q$ and $\mathbf{h}$ where $\mathbf{h}^Q$ is the histograms of the query image and $\mathbf{h}$ is the histogram of a subimage contained in a subwindow $w = [t,b,l,r]$. The χ^2 distance is defined as

$$\chi^2(\mathbf{h}^Q, \mathbf{h}) = \sum_{k=1}^{K} \frac{\{\mathbf{h}^Q(k) - \mathbf{h}(k)\}^2}{\mathbf{h}^Q(k) + \mathbf{h}(k)}. \tag{12}$$

Let $f_k(x) = -\{\mathbf{h}^Q(k) - x\}^2/\{\mathbf{h}^Q(k) + x\}$. Then we have

$$\begin{aligned} f_k'(x) &= -\frac{x^2 + 2x\mathbf{h}_Q(k) - 3\mathbf{h}_Q^2(k)}{\{x + \mathbf{h}_Q(k)\}^2} \\ f_k''(x) &= -\frac{8\mathbf{h}_Q^2(k)}{\{x + \mathbf{h}_Q(k)\}^3}. \end{aligned} \tag{13}$$

Note that $\mathbf{h}_Q(k) > 0$, it follows $f_k''(x) < 0$ for any $x \ge 0$ and therefore

$$f(\mathbf{h}) = \sum_k f_k\{\mathbf{h}(k)\} = -\chi^2(\mathbf{h}^Q, \mathbf{h}) \tag{14}$$

satisfies the condition in Theorem 1 and the associated 2D subarrays $A(\cdot,\cdot,l,r)$ (and $A(t,b,\cdot,\cdot)$) are inverse Monge matrices.

3.2. Euclidean Metric

The squared Euclidean distance of histograms $\mathbf{h}^Q$ and $\mathbf{h}$ is defined as

$$L^2(\mathbf{h}^Q, \mathbf{h}) = \sum_{k=1}^{K} (\mathbf{h}^Q(k) - \mathbf{h}(k))^2. \quad (15)$$

Let $f_k(x) = -(\mathbf{h}^Q(k) - x)^2$. Then we have $f_k''(x) = -1 \leq 0$ for any $x \geq 0$ and therefore

$$f(\mathbf{h}) = \sum_k f_k\{\mathbf{h}(k)\} = -L^2(\mathbf{h}^Q, \mathbf{h}) \quad (16)$$

satisfies the condition in Theorem 1 and the associated 2D subarrays $A(\cdot, \cdot, l, r)$ (and $A(t, b, \cdot, \cdot)$) are inverse Monge matrices.

3.3. Voting with Multiple Queries

Suppose we have multiple query images with histograms $\mathbf{h}_1^Q(k), \mathbf{h}_2^Q(k), \mathbf{h}_l^Q(k)$, we can assign weights $\alpha_i \geq 0$ for every query image and use the following quality function to detect objects:

$$f(\mathbf{h}) = \sum_{i=1}^{l} -\alpha_i D(\mathbf{h}_i^Q, \mathbf{h}) \quad (17)$$

where $D(\mathbf{h}_i^Q, \mathbf{h})$ is eitherχ^2 or Euclidean distance.

Similarly as the case with one query image, $f(\mathbf{h})$ can be decomposed as

$$f(\mathbf{h}) = \sum_k f_k\{\mathbf{h}(k)\} \quad (18)$$

where

$$f_k(x) = \sum_{i=1}^{l} -\alpha_i D(x, \mathbf{h}_i^Q(k)). \quad (19)$$

In Sections 3.1 and 3.2, we have shown that $D''(x, \mathbf{h}_i^Q(k)) \leq 0$ for any $x \geq 0$ where $D(x, y)$ is either χ^2 or Euclidean metric. Note that $\alpha_i \geq 0$, we have $f_k''(x) \leq 0$ for any $x \geq 0$ and therefore $f(\mathbf{h})$ satisfies the condition in Theorem 1 and the associated 2D subarrays $A(\cdot, \cdot, l, r)$ (and $A(t, b, \cdot, \cdot)$) are inverse Monge matrices.

4. Alternating Optimization

In [3], an alternating optimization method is presented for efficient subwindow search with linear decision function. Here, we adopt the same procedure but for nonlinear decision functions which satisfies the condition in Theorem 1 and therefore SMAWK algorithm can be applied in subwindow search, and we call this method Alternating Row and Column Search (ARCS). The ARCS method first initializes the row range as the full row range or initializes the column range as the full column range, and then applies the SMAWK algorithm to alternatingly optimize the column range and and the row range until convergence. More precisely, we start by initializing $[t : b] = [1 : m]$ and optimize the column interval $[l : r]$ by applying SMAWK algorithm. Then we fix the column interval and optimize the row interval $[t : b]$ again by applying the SMAWK algorithm. This alternating optimization of the row and column intervals is repeated until convergence. The convergence is guaranteed since the maximum entry of array A is bounded and the obtained entry is non-decreasing for each iteration. However, it may converge to a locally optimal subwindow instead of the global optimal solution.

Similarly, one can also start the algorithm by initializing the column interval $[l, r] = [1, n]$ rather than starting from the rows. We recommend to apply both initializations and choose the solution that yields the larger score. This can help reduce the problem of encountering a local maxima.

4.1. Complexity Analysis

The computational complexity of ESS and ARCS depend on the iterations. For an $n \times n$ images with $O(n^2)$ features in K cluster bins, the time complexity and memory requirement are summarized in Table 1.

Table 1. Time Complexity and Memory Requirement for $n \times n$ images with K cluster bins of features.

Methods	Time Complexity	Memory Requirement
ESS	$O(n^2K + L_{ess}K)$	$O(n^2K + L_{ess})$
ARCS	$O(L_{alt}(n^2 + nK))$	$O(n^2 + nK)$

The iteration number L_{ess} of ESS is typically of order $O(n^2)$ though it can be $O(n^4)$ in the worst case. The iteration number L_{alt} of ARCS is typically less than 10 in our experiments. And therefore ARCS is much faster than ESS. Also, since the upper bound of L_{ess} is $O(n^4)$, the performance variation of ESS is large across images.

The n^2K time and memory complexity of ESS comes from the computation and storage of the integral image for each histogram element. With these integral images, ESS takes $O(K)$ computations for each iteration. Hence, the total time and memory complexity are $O(n^2K + L_{ess}K)$ and $O(n^2K + L_{ess})$ respectively.

Since the iteration number L_{alt} of ARCS is much less than K, it is not beneficial to use integral images. Instead, we read the extracted features for each iteration and it takes $O(L_{alt}n^2)$. Combining the $O(nK)$ computations of SMAWK algorithm for each iteration, the total time complexity of ARCS is then $O(L_{alt}(n^2 + nK))$. For memory requirement, it includes the $O(n^2)$ features and a $n \times K$ matrix which stores the number of kth ($1 \leq k \leq K$)

features extracted in each row (or each column) when the column range (l, r) (or row range (t, b) respectively) is fixed. For the implementation of the SMAWK algorithm, one can use the public Python code which is available on *http://code.activestate.com/recipes/117244/*.

5. Application to Object Detection: A Two-Stage Learning Framework

In this section, we show an example of the applications of the proposed algorithms to object detection. For convenience, we call an image positive if the image contains the object for detection and call it negative otherwise. An object detector has two tasks: first it needs to find the correct object position if the object is in the image; second, it needs to give correct scores to the detected candidate object; that is, if the detected subimage is really the object, the score should be high; otherwise, the score should be low. Based on this observation, we propose to design separate quality functions for each task. The first one is called the localization quality function and it aims to localize the object correctly for the positive images. We train it based on the positive training images. The second one is called classification quality score and it aims to classify the subimages detected by the localization quality function.

5.1. Localization Quality Function

We choose the localization quality function as follows. First we compute the bag-of-words histograms $\mathbf{h}_i, i = 1, 2, \cdots, l$, of the objects contained in the positive training images based on the ground truth bounding box. Then we use $\mathbf{h}_i$ as the query image histogram and apply the optimum subwindow search algorithms to detect the optimal subwindow for each positive training image. By comparing the detected subwindow with the ground truth bounding box, we compute their matching rate with more than 50% overlap. The overlap of two windows w_1, w_2 is defined as the ratio of the common area $area\{w_1 \cap w_2\}$ and the union area $area\{w_1 \cup w_2\}$. We choose the $\mathbf{h}_i$ with maximal matching rate as the final query image histogram and apply the optimum subwindow search algorithms on the test images. In our experiments, we used the negative χ^2 distance between the test and query images as the quality function.

5.2. Classification Quality Function

With the detected subimages in both positive and negative images, the classification of these subimages is a standard classification problem and we apply least square support vector machine (LS-SVM) [18] to train the classifier. We use LS-SVM because the hyper parameters can be found easily by efficient cross-validation [2]. For clarity, we summarize LS-SVM briefly. Given a training set $\{(x_i, y_i)\}_{i=1}^n$ with input data $x_i \in \mathbb{R}^n$ and class labels $y_i \in \{-1, 1\}$, the classifier of LS-SVM [19, 18] takes the form

$$y(x) = \text{sign}\left[\sum_{i=1}^{n} \alpha_i K(x_i, x) + b\right]. \tag{20}$$

where $K(\cdot, \cdot)$ is the kernel function which can typically be either linear, polynomial or Gaussian kernels; α and b are the solution of the following linear equations

$$\begin{bmatrix} 0 & \mathbf{1}_n^T \\ \mathbf{1}_n & K + \frac{1}{\gamma} I_n \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix} \tag{21}$$

with $y = [y_1, y_2, \cdots, y_n], \mathbf{1}_n = [1, 1, \cdots, 1]^T, \alpha = [\alpha_1, \alpha_2, \cdots, \alpha_n]^T$.

In our application, x_i is the histogram of the detected subimage and y_i is the label. $y_i = 1$ if the detected image is a real object and the detected window is correct, that is, its overlap with the ground truth window is more than 50%. The kernel function is based on the χ^2 distance and we choose the Gaussian kernel

$$K(x_i, x_j) = exp\{-\chi^2(x_i, x_j)/\sigma^2\} \tag{22}$$

For the parameters σ and γ, we applied the efficient cross-validation method [2] to estimate the generalization performance and choose the parameters with minimal cross-validation errors. All the parameters selection are based on the training images.

6. Experiments

Our experiment is based on VLFeat[20], an open source library for feature extraction, and we test the performance on the PASCAL VOC 2006 database [10]. We used the fast dense feature extraction method [13] to extract the SIFT descriptors every 3 pixels along both rows and columns, and used the hierarchical K-means algorithm to cluster the features extracted from the first 100 training images into 1024 bins which formulates the bag-of-words vocabulary. The PASCAL VOC 2006 database consists of 5304 images containing 9507 objects from 10 categories. We choose two objects "cat" and "dog" to test our algorithm. For each object, we use objects in the training images to find a target histogram, and then apply the proposed algorithms to detect the optimal windows. The quality scores are given by a classifier which is trained based on the detected objects (instead of the ground truth objects) in the training images. All experiments [1] were conducted on a standard desktop PC (Intel Core 2 Quad 2.33GHz) running Windows XP, compiled using Visual Studio .NET 2005. No multi-threaded processing was utilized.

[1] Codes available at http://impca.cs.curtin.edu.au/downloads/software.php

6.1. Search Time Comparison

Table 2 and Table 3 report the search time of our proposed algorithm ARCS and ESS [14]. It shows that ARCS is significantly faster. In all cases, the run time of ARCS is within 1 seconds while ESS can take as long as 75 seconds. The iteration number of ARCS is typically less than 10 while ESS typically takes iterations of thousands. However, for each iteration, the computation of ARCS is heavier than ESS. Hence, when the iteration number of ESS is very small, say of hundreds, ESS can be faster than ARCS.

Table 2. Search Time Comparison for cat detection of PASCAL VOC 2006 on the 2686 test images. On average, ARCS is 27 time faster than ESS. ARCS stands for the alternating methods we proposed and ESS stands for the branch and bound method developed in [14]

	CPU Time (Seconds)		
Methods	Average	minimum	Maximum
ESS	3.628	0.092	75.49
ARCS	0.133	0.018	0.735

Table 3. Search Time Comparison for dog detection of PASCAL VOC 2006 on the 2686 test images. On average, ARCS is 28 time faster than ESS.

	CPU Time (Seconds)		
Methods	Average	minimum	Maximum
ESS	4.27	0.088	64.9
ARCS	0.148	0.017	0.730

6.2. Detection Performance Comparison

First, we compare the detection performance of ESS and ARCS. Both use our proposed two stage learning framework. The difference is that ESS finds global optimizer while ARCS may encounter local maxima problems. From Figures 1-2 and Table 4, one can see that their performances are almost identical.

Second, we compare the detection performance of the proposed algorithm to reported results on two objects: "cat" and "dog". The results are shown in Table 4. With a simple localization quality function and an LS-SVM classifier on the detected subimage, our two-stage learning object detector achieves better performance than [15] which used a linear SVM classifier. This shows that nonlinear object detector is better than linear methods and it is important to find efficient ways to evaluate the optimum subwindow search with nonlinear quality functions.

7. Conclusion

By using the Monge properties of the data structure of subwindow search in object detection, we have developed a fast locally optimum subwindow search algorithm. Also, we propose a two-stage learning framework for object detection in order to apply the proposed optimum subwindow search algorithm effectively. Experiments on the PASCAL-VOC 2006 data base demonstrates that the proposed algo-

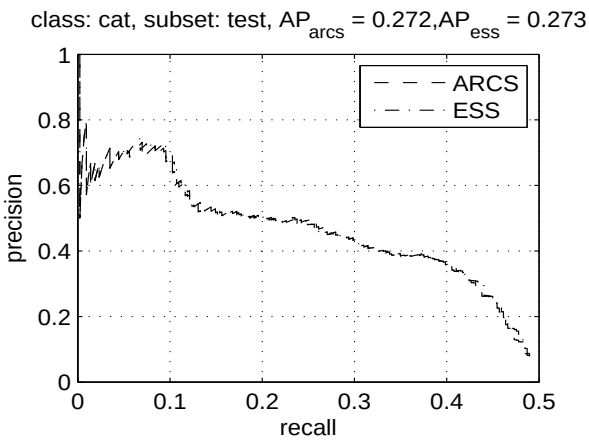

Figure 1. Performance Comparison of the Sub-Images Detected by ARCS and ESS on the Test Images for cat. ESS guarantees global optimization. Though ARCS can not guarantee the global optimization, the performance loss is negligible.

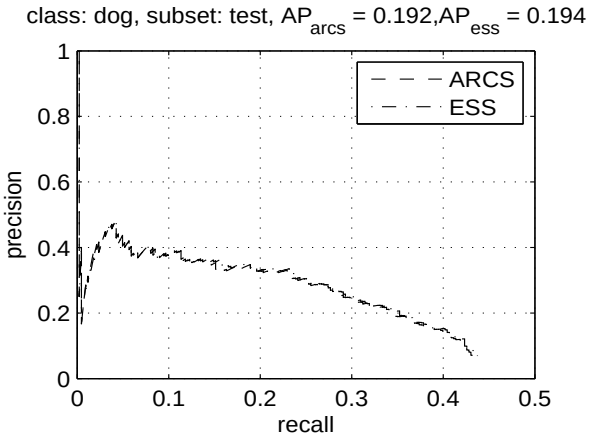

Figure 2. Performance Comparison of the Sub-Images Detected by ARCS and ESS on the Test Images for dog. ESS guarantees global optimization. Though ARCS can not guarantee the global optimization, the performance loss is negligible.

Table 4. Average Precision (AP) scores on the PASCAL-VOC 2006 data set. The performance of Shotton etal 2006 is the best results submitted to VOC 2006 challenge for these two objects.

Methods data set	cat	dog
Lampert etal 2009 [15]	0.223	0.148
Shotton etal 2006[10]	0.151	0.118
ARCS	**0.272**	**0.192**
ESS	**0.273**	**0.194**

rithm is significantly faster than the well-known efficient subwindow search methods. However, our fast search algorithm is only applicable for some special type of nonlinear quality functions which, unfortunately, do not include the wide-used nonlinear support vector machines. It needs further investigation to expand the cases of quality functions where Monge property holds and our algorithm can be applied.

Appendix: Proof of Theorems 1

Before we prove Theorem 1, we need consider a simple 1D subarray search problem and introduce a Lemma that is key to the proof of Theorem 1. Let $\{x_i\}_{i=1}^n$ be a non-negative 1D array, i.e., $x_i \geq 0$. Suppose we are interested to find a contiguous subarray so that its sum is closest to y among all possible contiguous subarrays. Let $d(x, y)$ denote the distance from x to y. Let $f(x) = -d(x, y)$. Then the problem can be formulated as

$$\max_{1\leq l\leq r\leq n} f\left(\sum_{i=l}^{r} x_i\right) \tag{23}$$

where l and r denote the left and right end of the subarray.

A naive solution of this problem is to try all the $n(n+1)/2$ possible subarrays and choose the one with minimal distance. We assume that

$$f''(x) \leq 0, \forall x \geq 0. \tag{24}$$

Let X denote an upper triangular matrix defined as

$$X_{ij} \triangleq \sum_{k=i}^{j} x_k, j \geq i. \tag{25}$$

Then we have the following Lemma

Lemma 1 *Let X be defined as in (25), and $f(x)$ satisfies the conditions (24). Then, for any $i < j < k < l$, we have*

$$f(X_{jl}) - f(X_{jk}) \geq f(X_{il}) - f(X_{ik}). \tag{26}$$

That is, $f(X)$ is an inverse Monge matrix.

Proof. First, we prove (26). From the definition of X_{ij} in (25), we have

$$X_{ik} - X_{jk} = X_{il} - X_{jl} = \sum_{s=i}^{j} x_s \tag{27}$$

and therefore

$$\begin{aligned} f(X_{il}) - f(X_{ik}) &= \int_{X_{jk}}^{X_{il}} f'(x)dx \\ &= \int_{X_{jk}+\delta}^{X_{jl}+\delta} f'(x)dx \\ &= \int_{X_{jk}}^{X_{jl}} f'(x+\delta)dx \\ &\leq \int_{X_{jk}}^{X_{jl}} f'(x)dx \\ &= f(X_{jl}) - f(X_{jk}) \end{aligned} \tag{28}$$

where $\delta = \sum_{s=i}^{j} x_s \geq 0$ and the inequality holds since $f'(x+\delta) \leq f'(x)$ (following (24)). The inequality (26) is proved.

□

Now we are ready to prove Theorem 1.

Proof of Theorem 1: Since $f(\mathbf{h}) = \sum_{k=1}^{K} f_k(\mathbf{h}(k))$, we have

$$A[t, b, l, r] = \sum_{k=1}^{K} A_k[t, b, l, r] \tag{29}$$

where

$$A_k[t, b, l, r] = \begin{cases} f_k(\mathbf{h}_w(k)); & \text{if } w = [t : b, l : r] \in \mathcal{W} \\ -\infty, & \text{otherwise} \end{cases} \tag{30}$$

Note that $f_k''(x) \leq 0$. From Lemma 1, it follows that $A_k[t, b, l, r]$ is an inverse Monge matrix. Since each $A_k[t, b, l, r]$ is an inverse matrix, its sum $A[t, b, l, r]$ is also an inverse matrix following Proposition 2 (in Section 2), and this concludes the proof.

□

References

[1] A. Aggarwal, M. M. Klawe, S. Moran, P. Shor, and R. Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2:195–208, 1987. 2

[2] S. An, W. Liu, and S. Venkatesh. Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression. *Pattern Recognition*, 40:2154–2162, 2007. 5

[3] S. An, P. Peursum, W. Liu, and S. Venkatesh. Efficient algorithms for subwindow search in object detection and localization. In *Proceedings of CVPR*, 2009. 1, 4

[4] M. J. Atallah and S. R. Kosaraju. An efficient parallel algorithm for the row minima of a totally monotone matrix. In *Proc. of the second annual ACM-SIAM symposium on discrete algorithms*, pages 394–403, 1991. 2

[5] H. Bay, T. Tuytelaars, and L. J. Gool. SURF: Speeded Up Robust Features. In *Proceedings of ECCV*, 2006. 1

[6] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *Proceedings of the 6th ACM international Conference on image and video retrieval*, pages 401–408, 2007. 1

[7] R. E. Burkard, B. Klinz, and R. Rudolf. Perspectives of Monge properties in optimization. *Discrete Applied Mathematics*, 70:95–161, 1996. 1, 2

[8] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *Proceedings of CVPR*, 2007. 1

[9] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of CVPR*, 2005. 1

[10] M. Everingham, A. Zisserman, C. K. I. Williams, and L. Van Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results. http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf. 5, 6

[11] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 30(1):36–51, 2008. 1

[12] M. Fritz and B. Schiele. Decomposition, discovery and detection of visual caregories using topic models. In *Proceedings of CVPR*, 2008. 1

[13] B. Fulkerson, A. Vedaldi, and S. Soatto. Localizing objects with smart dictionaries. In *Proceedings of ECCV*, 2008. 5

[14] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Proceedings of CVPR*, 2008. 1, 3, 6

[15] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Efficient subwindow search: a branch and bound framework for object localization. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2009. To Appear. 1, 6

[16] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 1

[17] H. Rowley, S. Baluja, and T. Kanade. Human face detection in visual scene. In *Advances in Neural Information Processing Systems (NIPS96)*, pages 875–881, 1996. 1

[18] J. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least squares support vector machines*. World Scientific, 2002. 5

[19] J. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9:293–300, 1999. 5

[20] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008. 5

[21] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2009. 1

[22] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings CVPR*, 2001. 1